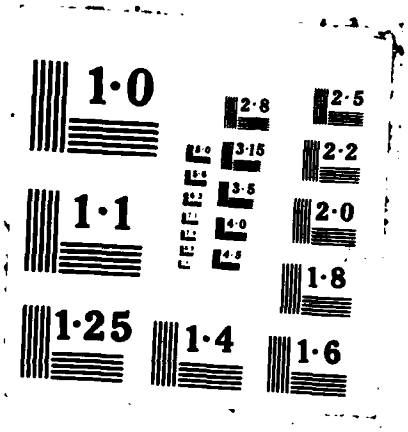
AD-A188 620 CHARACTERIZING KRIPKE STRUCTURES IN TEMPORAL LOGIC(U) 1/1
CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER
SCIENCE.. M C BROWNE ET AL. DEC 87 CMU-CS-87-104
UNCLASSIFIED AFWAL-TR-87-1168 F33615-84-K-1520 F/G 12/5 NL


1·0
2·8
2·5
5·0
3·15
5·6
2·2
6·3
3·5
7·1
2·0
8·0
4·0
9·0
4·5
1·1
1·8
1·25
1·4
1·6

PHOTOGRAPH THIS SHEET

AD-A188 620

DTIC ACCESSION NUMBER

LEVEL

INVENTORY

1

AFWAL-TR-87-1168

DOCUMENT IDENTIFICATION

Dec 1987

This document has been approved for public release and sale; its distribution is unlimited.

DISTRIBUTION STATEMENT

ACCESSION FOR

NTIS	GRA&I	☒
DTIC	TAB	□
UNANNOUNCED		□
JUSTIFICATION		

BY

DISTRIBUTION /

AVAILABILITY CODES

DIST	AVAIL AND/OR SPECIAL	
A-1		

DISTRIBUTION STAMP

DATE ACCESSIONED

DATE RETURNED

88 2 05 099

DATE RECEIVED IN DTIC

REGISTERED OR CERTIFIED NO.

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-DDAC

DOCUMENT PROCESSING SHEET

PREVIOUS EDITION MAY BE USED UNTIL STOCK IS EXHAUSTED.

AFWAL-TR-87-1168

CHARACTERIZING KRIPKE STRUCTURES IN TEMPORAL LOGIC

M.C. Browne, E.M. Clarke and O. Grumberg

Carnegie-Mellon University
Computer Science Department
Pittsburgh, PA 15213-3890

December 1987

Interim

AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6543

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

Chahira M. Hopper

CHAHIRA M. HOPPER
Project Engineer

Richard C Jones

RICHARD C. JONES
Ch, Advanced Systems Research Gp
Information Processing Technology Br

FOR THE COMMANDER

Edward L. Gliatti

EDWARD L. GLIATTI
Ch, Information Processing Technology Br
Systems Avionics Div

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/AAAT, Wright-Patterson AFB, OH 45433-6543 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No 0704-0188*

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CMU-CS-87-104	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFWAL-TR-87-1168

6a. NAME OF PERFORMING ORGANIZATION Carnegie-Mellon University	6b OFFICE SYMBOL *(If applicable)*	7a. NAME OF MONITORING ORGANIZATION Air Force Wright Aeronautical Laboratories AFWAL/AAAT-3
6c. ADDRESS *(City, State, and ZIP Code)* Computer Science Dept Pittsburgh PA 15213-3890		7b ADDRESS *(City, State, and ZIP Code)* Wright-Patterson AFB OH 45433-6543
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b OFFICE SYMBOL *(If applicable)*	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-84-K-1520

8c. ADDRESS *(City, State, and ZIP Code)*	10. SOURCE OF FUNDING NUMBERS			
	PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO
	61101E	4976	00	01

11. TITLE *(Include Security Classification)*

Characterizing Kripke Structures In Temporal Logic

12. PERSONAL AUTHOR(S)
M. C. Browne, E. M. Clarke, O. Grumberg

13a. TYPE OF REPORT Interim	13b TIME COVERED FROM ________ TO ______	14. DATE OF REPORT *(Year, Month, Day)* 1987 December	15 PAGE COUNT 23

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18 SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION

22a. NAME OF RESPONSIBLE INDIVIDUAL Chahira M. Hopper	22b TELEPHONE *(Include Area Code)* (513) 255-7865	22c OFFICE SYMBOL AFWAL/AAAT-3

Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

TABLES OF CONTENTS

LIST OF FIGURES

Characterizing Kripke Structures in Temporal Logic

M. C. Browne

E. M. Clarke

O. Grümberg
Carnegie Mellon University, Pittsburgh

1. Introduction

The question of whether branching-time temporal logic or linear-time temporal logic is best for reasoning about concurrent programs is one of the most controversial issues in logics of programs. Concurrent programs are usually modelled by labelled state-transition graphs in which some state is designated as the initial state. For historical reasons such graphs are called Kripke structures ~~[8].~~ In linear temporal logic, operators are provided for describing events along a single time path (i.e., along a single path in a Kripke structure). In a branching-time logic the temporal operators quantify over the futures that are possible from a given state (i.e., over the possible paths that lead from a state). It is well known that the two types of temporal logic have different expressive powers ~~([4], [9])~~. Linear temporal logic, for example, can express certain *fairness* properties that cannot be expressed in branching-time temporal logic. On the other hand, certain practical decision problems like *model checking* ~~([3], [16])~~ are easier for branching-time temporal logic than for linear temporal logic.

It is shown

~~In~~ this paper ~~we~~ provide further insight on which type of logic is best. ~~We show~~ that if two *finite* Kripke structures can be distinguished by some formula that contains both branching-time and linear-time operators, then the structures can be distinguished by a formula that contains only branching time operators. Specifically, we show that if two finite Kripke structures can be distinguished by some formula of the logic CTL* (i.e., if there is some CTL* formula that is true in one but not in the other), then they can be distinguished by some formula of the logic CTL. The logic CTL* ([3], [4]) is a very powerful temporal logic that combines both branching-time and linear-time operators; a path quantifier, either A ("for all paths") or E ("for some paths") can prefix an assertion composed of arbitrary combinations of the ususal linear time operators G ("always"), F ("sometimes"), X ("nexttime"), and U ("until"). CTL ([1], [2]) is a restricted subset of CTL* that permits only branching-time operators--each path quantifier must be immediately followed by exactly one of the operators G, F, X, or U.

Our goal is to show that for any finite Kripke structure M, it is possible to construct a CTL formula F_M that

This research was partially supported by NSF Grant MCS-82-16706. The third author, O Grumberg, is currently on leave from Technion, Haifa and is partially supported by a Weizmann postdoctoral fellowship.

uniquely characterizes M. Since one Kripke structure may be a trivial unrolling of another, we use a notion of equivalence between Kripke structures that is similar to the notion of *bisimulation* studied by Milner [12]. We say that states s and s' are *equivalent* if they have the same labelling of atomic propositions and for each transition from one of the two states to some state t there is a corresponding transition from the other state to a state t' that is equivalent to t. Two Kripke structures are equivalent if their initial states are equivalent. It is not difficult to prove that if two Kripke structures are equivalent, then their initial states must satisfy the same CTL*.

An obvious first attempt to construct F_M is simply to write a CTL formula that specifies the transition relation of M. For each state s in M we include in F_M a conjunct of the form

$$\mathrm{AG}(\mathcal{L}(s) \Rightarrow \bigwedge_i \mathrm{EX}\mathcal{L}(s_i) \wedge \mathrm{AX}(\bigvee_i \mathcal{L}(s_i)))$$

where $s_1, \ldots, s_n$ are the successors of s and $\mathcal{L}(t)$ is the labelling of atomic propositions associated with state t. It is easy to see, however, that this simple approach cannot work in general: several states in M may have exactly the same labelling of atomic propositions.

Instead, we first show that it is possible to write a CTL formula that will distinguish between two states in the same structure that are not equivalent according to the above definition. Two inequivalent states may have exactly the same labelling of atomic propositions, they may even have corresponding successors, but the computation trees rooted at those states must differ at some finite depth. The difference in the computation trees can be exploited to give a CTL formula that distinguishes between the states. Since equivalent states satisfy the same CTL* formulas, it follows that if two states can be distinguished by a CTL* formula, they can be distinguished by a CTL formula. Once we can distinguish between inequivalent states in the same structure, we can write a single CTL formula that encodes the entire Kripke structure; this formula is the F_M that we seek.

The above construction requires the use of the nexttime operator in specifying F_M. In reasoning about concurrent systems, however, the nexttime operator may be dangerous, since it refers to the *global* next state instead of the *local* next state within a process [10]. What happens if we disallow the nexttime operator in CTL formulas? The proof, in this case, requires another notion of equivalence--*equivalence with respect to stuttering*. We say that two state sequences *correspond* if each can be partitioned into finite blocks of identically labelled states such that each state in the i-th block in one sequence is equivalent to each state in the i-th block of the other sequence. Thus, duplicating some state in a sequence any finite number of times will always result in a corresponding sequence. We say that two states are *equivalent* if for each state sequence starting at one there is a corresponding state sequence that starts at the other. Under this second notion of equivalence the proof of the characterization theorem becomes much more complicated, since it is possible for two inequivalent states to have exactly the same finite behaviors (modulo stuttering), but different infinite behaviors.

Equivalence under stuttering turns out to be quite useful for reasoning about hierarchically constructed concurrent systems. In determining the correctness of such a system by using a technique like temporal logic model checking ([2], [3], [11], [13], [16], [17]), it is often desirable to replace a low level module by an equivalent structure with fewer states. Our results show how this can be done while preserving all of those properties that are invariant under stuttering. We give polynomial algorithms both for determining if two structures are equivalent with respect to stuttering and for minimizing the number of states in a given structure under this notion of equivalence.

Finally, our results have some interesting implications for the problem of synthesizing finite state concurrent systems from temporal logic specifications ([2], [14]). In order to guarantee that any Kripke structure can be synthesized from a specification in linear temporal logic, Wolper [18] was forced to introduce more complicated operators based on regular expressions. Our results show that (at least in theory) no such extension is necessary for branching-time temporal logic. Any Kripke structure can be specified directly by a formula of branching-time logic.

The expressive power of various temporal logics has been discussed in several papers; see ([4], [9]) for example. Hennessy and Milner [7], Graf and Sifakis [6], and Pnueli [15] have all discussed the relationship between temporal logic and various notions of equivalence between models of concurrent programs. However, we believe that we are the first to show that it is possible to characterize Kripke models within branching-time logic and to investigate the consequences of this result.

Our paper is organized as follows: In Section 2 we describe the logics CTL and CTL^*. In Section 3, we state formally what it means for two states in a Kripke structure to be equivalent and prove that equivalent states satisfy exactly the same CTL^* formulas. Section 3 also contains the first of the two main results of the paper: we show how to characterize Kripke structures using CTL formulas with the nexttime operator. Section 4 introduces the second notion of equivalence (equivalence with respect to stuttering) and shows that if the nexttime operator is disallowed, then equivalent states again satisfy exactly the same CTL^* formulas. We also extend the characterization theorem of Section 3 to Kripke structures with the new notion of equivalence. In Section 5 we give a polynomial algorithm for determining if two states are equivalent up to stuttering. The paper concludes in Section 6 with a discussion of some remaining open problems like the possibility of extending our results to Kripke structures with fairness constraints (i.e., Büchi Automata).

2. The Logics CTL and CTL*

There are two types of formulas in CTL*: *state formulas* (which are true in a specific state) and *path formulas* (which are true along a specific path). Let AP be the set of atomic proposition names. A state formula is either:

- A, if $A \in AP$.
- If f and g are state formulas, then $\neg f$ and $f \vee g$ are state formulas.
- If f is a path formula, then $\mathrm{E}(f)$ is a state formula.

A path formula is either:

- A state formula.
- If f and g are path formulas, then $\neg f$, $f \vee g$, $\mathrm{X}\, f$, and $f\, \mathrm{U}\, g$ are path formulas

CTL* is the set of state formulas generated by the above rules

CTL is a subset of CTL* in which we restrict the path formulas to be

- If f and g are state formulas, then $\mathrm{X}\, f$ and $f\, \mathrm{U}\, g$ are path formulas
- If f is a path formula, then so is $\neg f$.

We define the semantics of both logics with respect to a structure $M = \langle S, R, \mathcal{L} \rangle$, where

- S is a set of states.
- $R \subseteq S \times S$ is the transition relation, which must be total. We write $s_1 \rightarrow s_2$ to indicate that $(s_1, s_2) \in R$.
- $\mathcal{L}: S \rightarrow \mathcal{P}(AP)$ is the proposition labeling.

Unless otherwise stated, all of our results apply only to finite Kripke structures.

We only consider transition relations where every state is reachable from the initial state. We define a *path in M* to be a sequence of states, $\pi = s_0, s_1, \ldots$ such that for every $i \geq 0$, $s_i \rightarrow s_{i+1}$. π^i will denote the *suffix* of π starting at s_i.

We use the standard notation to indicate that a state formula f holds in a structure: $M, s \models f$ means that f holds at state s in structure M. Similarly, if f is a path formula, $M, \pi \models f$ means that f holds along path π in structure M. The relation $\models$ is defined inductively as follows (assuming that f_1 and f_2 are state formulas and g_1 and g_2 are path formulas):

1. $s \models A \quad \Leftrightarrow \quad A \in \mathcal{L}(s)$.
2. $s \models \neg f_1 \quad \Leftrightarrow \quad s \not\models f_1$.
3. $s \models f_1 \vee f_2 \quad \Leftrightarrow \quad s \models f_1$ or $s \models f_2$.

4. $s \models \mathrm{E}(g_1)$ $\Leftrightarrow$ there exists a path π starting with s such that $\pi \models g_1$.

5. $\pi \models f_1$ $\Leftrightarrow$ s is the first state of π and $s \models f_1$.

6. $\pi \models \neg g_1$ $\Leftrightarrow$ $\pi \not\models g_1$.

7. $\pi \models g_1 \vee g_2$ $\Leftrightarrow$ $\pi \models g_1$ or $\pi \models g_2$.

8. $\pi \models \mathrm{X} g_1$ $\Leftrightarrow$ $\pi^1 \models g_1$.

9. $\pi \models g_1 \mathrm{U} g_2$ $\Leftrightarrow$ there exists a $k \geq 0$ such that $\pi^k \models g_2$ and for all $0 \leq j < k$, $\pi^j \models g_1$.

We will also use the following abbreviations in writing CTL* (and CTL) formulas:

- $f \wedge g \equiv \neg(\neg f \vee \neg g)$
- $\mathrm{F} f \equiv \mathit{true}\, \mathrm{U} f$
- $\mathrm{A}(f) \equiv \neg \mathrm{E}(\neg f)$
- $\mathrm{G} f \equiv \neg \mathrm{F} \neg f$.

3. Equivalence of Kripke Structures

Given two structures M and M' with the same set of atomic propositions AP, we define a sequence of equivalence relations $E_0, E_1, \ldots$ on $S \times S'$ as follows:

- $s E_0 s'$ if and only if $\mathcal{L}(s) = \mathcal{L}(s')$.
- $s E_{n+1} s'$ if and only if
  - $\mathcal{L}(s) = \mathcal{L}(s')$,
  - $\forall s_1 [s \rightarrow s_1 \Rightarrow \exists s_1' [s' \rightarrow s_1' \wedge s_1 E_n s_1']]$, and
  - $\forall s_1' [s' \rightarrow s_1' \Rightarrow \exists s_1 [s \rightarrow s_1 \wedge s_1 E_n s_1']]$.

Now, we define our notion of *equivalence between states*: sEs' if and only if sE_is' for all $i \geq 0$. Furthermore, we say that M with initial state s_0 is equivalent to M' with initial state s_0' iff $s_0 E s_0'$.

Lemma 1: Let sEs', then for every path, $s, s_1, \ldots$, there exists a path, $s', s_1', \ldots$ such that $\forall i [s_i E s_i']$.

Proof: Note first that $E_{n+1} \subseteq E_n$. Since E_0 is finite, there must be a k such that $E_{k+1} = E_k = E$. Thus, we can substitute E for E_k in the definition of E_{k+1} giving sEs' if and only if

- $\mathcal{L}(s) = \mathcal{L}(s')$,
- $\forall s_1 [s \rightarrow s_1 \Rightarrow \exists s_1' [s' \rightarrow s_1' \wedge s_1 E_n s_1']]$, and
- $\forall s_1' [s' \rightarrow s_1' \Rightarrow \exists s_1 [s \rightarrow s_1 \wedge s_1 E_n s_1']]$.

The remainder of the proof is a straightforward induction on the length of the path. □

Theorem 2: If sEs', then $\forall f \in CTL^{*}[s \models f \Leftrightarrow s' \models f]$.

This theorem is a consequence of the following lemma:

Lemma 3: Let h be either a state formula or a path formula. Let $\pi = s,s_1,\ldots$ be a path in M and $\pi' = s',s_1',\ldots$ be a path in M' such that sEs' and $\forall i\,[s_i E s_i']$. Then
$s \models h \Leftrightarrow s' \models h$, if h is a state formula and
$\pi \models h \Leftrightarrow \pi' \models h$, if h is a path formula.

Proof: We prov.: the theorem by induction on the structure of h.

Base: $h = A$. By the definition of E, $s \models A \Leftrightarrow s' \models A$.

Induction: There are several cases.

1. $h = \neg h_1$, a state formula.

$$\begin{aligned} s \models h &\Leftrightarrow s \not\models h_1 \\ &\Leftrightarrow s' \not\models h_1 \text{ (induction hypothesis)} \\ &\Leftrightarrow s' \models h \end{aligned}$$

The same reasoning holds if h is a path formula.

2. $h = h_1 \vee h_2$, a state formula.

Without loss of generality,

$$\begin{aligned} s \models h &\Leftrightarrow s \models h_1 \text{ or } s \models h_2 \\ &\Rightarrow s \models h_1 \\ &\Leftrightarrow s' \models h_1 \text{ (induction hypothesis)} \\ &\Rightarrow s' \models h \end{aligned}$$

The argument is the same in the other direction. We can also use this argument if h is a path formula.

3. $h = \mathrm{E}(h_1)$, a state formula.

Suppose that $s \models h$. Then there is a path, π_1 starting with s such that $\pi_1 \models h_1$. By Lemma 1, there is a corresponding path π_1' in M' starting with s'. So by the induction hypothesis, $\pi_1 \models h_1 \Leftrightarrow \pi_1' \models h_1$. Therefore, $s \models \mathrm{E}(h_1) \Rightarrow s' \models \mathrm{E}(h_1)$. We can use the same argument in the other direction, so the lemma holds.

4. $h = h_1$, where h is a path formula and h_1 is a state formula.

Although the lengths of h and h_1 are the same, we can imagine that h = path(h_1), where path is an operator which converts a state formula into a path formula. Therefore, we are simplifying h by dropping this path operator. So now:

$$\begin{aligned} \pi \models h &\Leftrightarrow s \models h_1 \\ &\Leftrightarrow s' \models h_1 \text{ (induction hypothesis)} \\ &\Rightarrow \pi' \models h. \end{aligned}$$

The reverse direction is similar.

5. $h = \mathrm{X}\, h_1$, a path formula.

By the definition of the next-time operator, $\pi^1 \models h_1$. Since π and π' correspond, so do π^1 and π'^1. Therefore, by the inductive hypothesis, $\pi'^1 \models h_1$ so $\pi' \models h$.

We can use the same argument in the other direction.

6. $h = h_1 \mathrm{U}\, h_2$, a path formula.

Suppose that $\pi \models h_1 \mathrm{U}\, h_2$. By the definition of the until operator, there is a k such that $\pi^k \models h_2$ and for all $0 \le j < k$, $\pi^j \models h_1$. Since π and π' correspond, so do π^j and π'^j for any j. Therefore, by the inductive hypothesis, $\pi'^k \models h_2$ and $\pi'^j \models h_1$ for all $0 \le j < k$. Therefore $\pi' \models h$.

We can use the same argument in the other direction. □

Another property of two equivalent states is that they both have corresponding *computation trees*. For every $s \in S$, $\mathrm{Tr}_n(s)$ is the computation tree of depth n rooted at s. Formally, $\mathrm{Tr}_0(s)$ consists of a single node which has the same label as s. $\mathrm{Tr}_{n+1}(s)$ has as its root a node m with the same label as s. If s has successors $s_1, \ldots, s_p$ in the Kripke structure, then node m will have subtrees $\mathrm{Tr}_n(s_1)$, ..., $\mathrm{Tr}_n(s_p)$.

Two trees $\mathrm{Tr}_n(s)$ and $\mathrm{Tr}_n(s')$ *correspond* (denoted $\mathrm{Tr}_n(s) \equiv \mathrm{Tr}_n(s')$) if and only if both of their roots have the same label and for every subtree of depth $n-1$ of the root of one, it is possible to find a corresponding subtree of the root of the other.

Lemma 4: $s E_n s'$ if and only if $\mathrm{Tr}_j(s) \equiv \mathrm{Tr}_j(s')$ for all $j \le n$.

Lemma 5: Given a *finite* set of states $s_1, \ldots, s_n$ there exists a c such that if two states s_i and s_j are not E-equivalent then $\mathrm{Tr}_c(s_i)$ and $\mathrm{Tr}_c(s_j)$ will not correspond.

We will call the value of c for S the *characteristic number* of the structure.

We associate a CTL formula with a tree $\mathrm{Tr}_n(s)$ as follows:

- $\mathcal{F}[\mathrm{Tr}_0(s)] = (p_1 \wedge \ldots \wedge p_u) \wedge (\neg q_1 \wedge \ldots \neg q_v)$, where $\mathcal{L}(s) = \{p_1, \ldots, p_u\}$ and $AP - \mathcal{L}(s) = \{q_1, \ldots, q_v\}$.
- $\mathcal{F}[\mathrm{Tr}_{n+1}(s)] = \left(\bigwedge_i \mathrm{EX}\, \mathcal{F}[\mathrm{Tr}_n(s_i)]\right) \wedge \mathrm{AX}\left(\bigvee_i \mathcal{F}[\mathrm{Tr}_n(s_i)]\right) \wedge \mathcal{F}[\mathrm{Tr}_0(s)]$, where s_i is a successor of s.

Lemma 6: $s \models \mathcal{F}[\mathrm{Tr}_n(s)]$ for all $n \geq 0$.

Lemma 7: If $s \models \mathcal{F}[\mathrm{Tr}_n(s')]$, then $\mathrm{Tr}_n(s) \equiv \mathrm{Tr}_n(s')$.

Proof: The proof is by induction on n. The basis case is trivial. Thus, we assume that $n > 0$. Let $s_1, s_2, \ldots, s_p$ be the sons of s in $\mathrm{Tr}_n(s)$ and $s'_1, s'_2, \ldots, s'_q$ be the sons of s' in $\mathrm{Tr}_n(s')$.

It is easy to see that s and s' have the same labelling of atomic propositions.

We must show that $\mathrm{Tr}_{n-1}(s_{i_0})$ corresponds to some $\mathrm{Tr}_{n-1}(s'_j)$. Since $s \models \mathcal{F}[\mathrm{Tr}_n(s')]$, $s \models \mathrm{AX}\left(\bigvee_j \mathcal{F}[\mathrm{Tr}_{n-1}(s'_j)]\right)$. Since s_{i_0} is a successor of s, $s_{i_0} \models \mathcal{F}[\mathrm{Tr}_{n-1}(s'_{j_0})]$ for some j_0. Hence, $\mathrm{Tr}_{n-1}(s_{i_0}) \equiv \mathrm{Tr}_{n-1}(s'_{j_0})$ by our inductive hypothesis.

Finally, we must show that $\mathrm{Tr}_{n-1}(s'_{j_0})$ corresponds to some $\mathrm{Tr}_{n-1}(s_i)$. Since $s \models \mathcal{F}[\mathrm{Tr}_n(s')]$, $s \models \bigwedge_j \mathrm{EX}\mathcal{F}[\mathrm{Tr}_{n-1}(s'_j)]$. Since s'_{j_0} is a successor of s', $s \models \mathrm{EX}\mathcal{F}[\mathrm{Tr}_{n-1}(s'_{j_0})]$. Therefore, there exists an i_0 such that $s_{i_0} \models \mathcal{F}[\mathrm{Tr}_{n-1}(s'_{j_0})]$. Hence, $\mathrm{Tr}_{n-1}(s_{i_0}) \equiv \mathrm{Tr}_{n-1}(s'_{j_0})$ by our inductive hypothesis. □

Lemma 8: If s is a state in a Kripke structure M, then there is a CTL formula, $\mathfrak{C}(M,s)$ that determines s up to E-equivalence within M, i.e. $\mathfrak{C}(M,s)$ is true in s and every state in M that is E-equivalent to s but false in every state in M that is not equivalent to s.

Proof: We choose $\mathfrak{C}(M,s) = \mathcal{F}[\mathrm{Tr}_c(s)]$ where c is the characteristic number of M. $\mathfrak{C}(M,s)$ is true in s and hence in all states E-equivalent to s. Let s' be a state that is not E-equivalent to s; then $\mathrm{Tr}_c(s) \not\equiv \mathrm{Tr}_c(s')$. Hence, by lemma 7, $s' \not\models \mathfrak{C}(M,s)$. □

Theorem 9: Given a Kripke structure M with initial state s_0, there is a CTL formula $F(M,s_0)$ that characterizes that structure up to E-equivalence, i.e. $M', s'_0 \models F(M,s_0) \Leftrightarrow s_0 E s'_0$.

Proof: For any state s in M, let $s_1, \ldots, s_p$ be the successors of s. We define

$$G(M,s) = \mathrm{AG}(\mathcal{C}(M,s) \Rightarrow \bigwedge_i \mathrm{EX}\ \mathcal{C}(M,s_i) \wedge \mathrm{AX} \bigvee_i \mathcal{C}(M,s_i))$$

$G(M,s)$ describes all of the possible transitions from s. $F(M,s_0)$ is the formula $\mathcal{C}(M,s_0) \wedge \bigwedge_s G(M,s)$. If two structures M,s_0 and M',s_0' are equivalent then by theorem 2 they satisfy the same formulas. Since $M,s_0 \models F(M,s_0)$, so does M',s_0'.

For the other direction we show by induction on n that if $M',s_0' \models F(M,s_0)$ then $\mathrm{Tr}_n(s_0) \equiv \mathrm{Tr}_n(s_0')$ for all $n \geq 0$. By lemma 4, the two structures are then E-equivalent. □

Corollary 10: Given two structures M and M' with initial states s_0 and s_0' respectively, $s_0 E s_0'$ if and only if $\forall f \in CTL^*[M,s_0 \models f \Leftrightarrow M',s_0' \models f]$.

Corollary 11: Given two structures M and M' with initial states s_0 and s_0' respectively, if there is a formula of CTL* that is true in one and false in the other, then there is also a formula of CTL that is true in the one and false in the other.

We illustrate our method of characterizing Kripke structures with the example in figure 3-1.

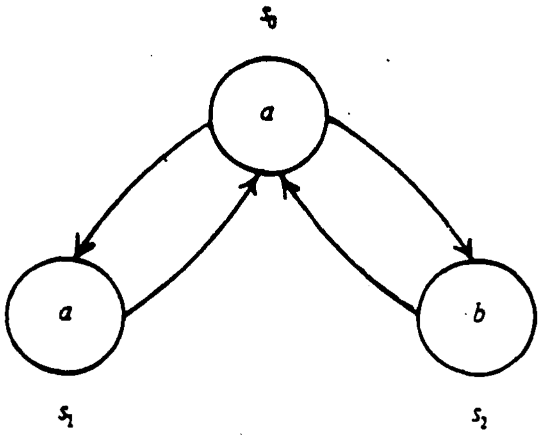


Figure 3-1: A Kripke structure in which every other state is labelled A

The characteristic number of this structure is 1, since $\mathrm{Tr}_0(s_0) \not\equiv \mathrm{Tr}_0(s_2)$, $\mathrm{Tr}_0(s_1) \not\equiv \mathrm{Tr}_0(s_2)$, and $\mathrm{Tr}_1(s_0) \not\equiv \mathrm{Tr}_1(s_1)$. Let

- $\mathcal{C}(M,s_0) = a \wedge \neg b \wedge \mathrm{EX}(a \wedge \neg b) \wedge \mathrm{EX}(\neg a \wedge b) \wedge \mathrm{AX}(a \wedge \neg b \vee \neg a \wedge b)$
- $\mathcal{C}(M,s_1) = a \wedge \neg b \wedge \mathrm{EX}(a \wedge \neg b) \wedge \mathrm{AX}(a \wedge \neg b)$

- $\mathcal{C}(M,s_2) = \neg a \wedge b \wedge \mathrm{EX}(a \wedge \neg b) \wedge \mathrm{AX}(a \wedge \neg b)$

We can now state the formula that characterizes this structure:

$$
\begin{aligned}
\mathcal{F}(M,s_0) = {} & \mathcal{C}(M,s_0) \wedge \\
& \mathrm{AG}(\mathcal{C}(M,s_0) \Rightarrow \mathrm{EX}\mathcal{C}(M,s_1) \wedge \mathrm{EX}\mathcal{C}(M,s_2) \wedge \mathrm{AX}(\mathcal{C}(M,s_1) \vee \mathcal{C}(M,s_2))) \wedge \\
& \mathrm{AG}(\mathcal{C}(M,s_1) \Rightarrow \mathrm{EX}\mathcal{C}(M,s_0) \wedge \mathrm{AX}\mathcal{C}(M,s_0)) \wedge \\
& \mathrm{AG}(\mathcal{C}(M,s_2) \Rightarrow \mathrm{EX}\mathcal{C}(M,s_0) \wedge \mathrm{AX}\mathcal{C}(M,s_0))
\end{aligned}
$$

4. Equivalence With Respect To Stuttering

We first define what it means for two Kripke structures to be-equivalent with respect to stuttering. Given two structures M and M' with the same set of atomic propositions, we define a sequence of equivalence relations $E_0, E_1, \ldots$ on $S \times S'$ as follows:

- $s E_0 s'$ if and only if $\mathcal{L}(s) = \mathcal{L}(s')$.
- $s E_{n+1} s'$ if and only if

  1. for every path π in M that starts in s there is a path π' in M' that starts in s', a partition $B_1 B_2 \ldots$ of π, and a partition $B_1' B_2' \ldots$ of π' such that for all $j \in \mathbb{N}$, B_j and B_j' are both non-empty and finite, and every state in B_j is E_n-related to every state in B_j', and
  2. for every path π' in M' starting in s' there is a path π in M starting in s that satisfies the same condition as in 1.

Our notion of *equivalence with respect to stuttering* is defined as follows: $s E s'$ if and only if $s E_i s'$ for all $i \geq 0$. Furthermore, we say that M with initial state s_0 is equivalent to M' with initial state s_0' if $s_0 E s_0'$.

Lemma 12: Given two Kripke structures M and M', there exists an l such that $\forall s \forall s' \, [s E_l s' \text{ iff } s E s']$.

Proof: By the definition of E_{l+1}, $sE_{l+1}s' \Rightarrow sE_l s'$, so $E_0 \supseteq E_1 \supseteq E_2 \ldots$. Since M and M' are both finite, E_0 must be finite as well, so only a finite number of these containments can be proper. Let E_l be the last relation that is properly included in E_{l-1}. By the definition of proper containment, $\forall m \geq l \, [E_l = E_m]$, so $sE_l s' \Rightarrow sE_m s'$, for $m \geq l$. Since $sE_l s' \Rightarrow sE_{l-1} s' \Rightarrow sE_{l-2} s' \ldots$, we have $sE_l s' \Rightarrow \forall m \, [sE_m s']$, so $sE_l s' \Rightarrow sEs'$. The other direction is trivial. □

Theorem 13: If sEs', then for every CTL* formula f without the nexttime operator, $s \models f$ iff $s' \models f$.

The proof is similar to that of theorem 2.

Lemma 14: Given a Kripke structure M, for every state $s \in M$, there is a CTL formula $\mathcal{C}(M,s)$ such that $\forall t \in M \, [t \models \mathcal{C}(M,s) \text{ iff } sEt]$.

Proof: We will prove by induction on l:

- If $\neg(sE_l t)$, then there is a CTL formula $d_l(s,t)$ such that $\forall v \in M[sE_l v \Rightarrow v \models d_l(s,t)]$ and $t \not\models d(s,t)$.

- For every state $s \in M$, there is a CTL formula $\mathbb{C}_l(M,s)$ such that for every $t \in M$, $t \models \mathbb{C}_l(M,s)$ iff $sE_l t$.

$d_l(s,t)$ is a formula that distinguishes between t and states equivalent to s within the structure M, and $\mathbb{C}_l(M,s)$ is a formula that characterizes E_l-equivalence to state s within M.

If we let $\mathbb{C}_l(M,s)$ be a conjunction of $\mathbb{C}_{l-1}(M,s)$ and $d_l(s,t)$ for every t that is not E_l-related to s, the second assertion follows easily. By lemma 12, this condition implies that the lemma is true. Now it is necessary to show how to construct $d_l(s,t)$ by induction on l.

Basis ($l=0$): Let $\{p_i\}$ be the set of atomic propositions in $\mathcal{L}(s)$ and $\{q_j\}$ be the set of atomic propositions in $AP-\mathcal{L}(s)$. Now, let

$$\mathbb{C}_0(M,s) = d_0(s,t) = \bigwedge_i p_i \wedge \bigwedge_j \neg q_j$$

It is clear that this formula is only true in states with the same labelling of atomic propositions as s. Therefore, the base case is established.

Induction: Assume that the result is true for l. We will show it for $l+1$.

Since $\neg(sE_{l+1}t)$, either there is a path from s without a corresponding path from t, or vice versa. In the latter case, we will use the argument below to find a $d_{l+1}(t,s)$ such that $t \models d_{l+1}(t,s)$ and $s \not\models d_{l+1}(t,s)$. We can negate this formula to get the desired $d_{l+1}(s,t)$.

If there is a path from s without a corresponding path from t, we can divide this path into blocks $(B_1 B_2 \ldots)$ such that:

$$\forall i\,[x \in B_i \Rightarrow x \models \mathbb{C}_l(M,\text{first}(B_i)) \text{ and } \text{first}(B_{i+1}) \not\models \mathbb{C}_l(M,\text{first}(B_i))].$$

Now, there are two cases: either there is a finite path from one state without a corresponding path from the other, or there is an infinite path without a corresponding path, but every finite prefix of this path has a corresponding path.

In the first case, the path from s is finite, so the blocks are finite and there are only a finite number of them (say n). Consider the CTL formula:

$$d_{l+1}(s,t) = \mathbb{C}_l(M,\text{first}(B_1)) \wedge \mathbf{E}[\mathbb{C}_l(M,\text{first}(B_1))\ \mathbf{U}\ \mathbb{C}_l(M,\text{first}(B_2)) \wedge \mathbf{E}[\ldots \mathbf{U}\ \mathbb{C}_l(M,\text{first}(B_n))]\ldots]$$

It is clear that $s \models d_{l+1}(s,t)$ along the path $B_1 B_2 \ldots B_n$. However, if $t \models d_{l+1}(s,t)$ then there is a path that can be partitioned into blocks $B'_1 B'_2 \ldots B'_n$ such that $\forall i\,[v \in B'_i \Rightarrow v \models \mathbb{C}_l(M,\text{first}(B_i))]$. Since every state in B_i

satisfies $\mathfrak{C}_l(M,\text{first}(B_i))$, the inductive hypothesis and the definition of E_l gives $B_i E_l B'_i$. Therefore, this path from t corresponds to the path from s, a contradiction. We conclude that $t \nvDash d_{l+1}(s,t)$.

In the second case, we start by showing that the path from s has only a finite number of blocks by using an argument based on König's lemma. We can construct a tree rooted at t such that $tt_1 \ldots t_n$ is a path through the tree if and only if there is a path in the Kripke structure $tu_1 \ldots u_p t_1 v_1 \ldots v_q t_2 \ldots t_n$ that corresponds to a prefix of the path from s with $B'_1 = \langle tu_1 \ldots u_p \rangle$, $B'_2 = \langle t_1 v_1 \ldots v_q \rangle$, and so on. Now, if the path from s has an infinite number of blocks, this tree must have an infinite number of nodes. Otherwise, if the tree had n nodes, there could be no path of length $n+1$, so the first $n+1$ blocks of the path from s would have no corresponding path from t. Since the Kripke structure is finite, we also know that this tree must be finitely branching. Therefore, by König's lemma, there must be an infinite path through the tree. But this implies that there is an infinite path from t that can be divided into an infinite number of blocks that correspond to the blocks of the path from s, so there is a path from t corresponding to the path from s, violating our assumption. Therefore, the path from s has only a finite number of blocks.

So, suppose that there are n blocks, all of which are finite except the last. Consider the CTL formula:
$d_{l+1}(s,t) = \mathfrak{C}_l(M,\text{first}(B_1)) \wedge \text{E}[\mathfrak{C}_l(M,\text{first}(B_1)) \text{ U } \mathfrak{C}_l(M,\text{first}(B_2)) \wedge \text{E}[\ldots \text{ U EG } \mathfrak{C}_l(M,\text{first}(B_n))] \ldots]$
It is clear that $s \vDash d_{l+1}(s,t)$ along the path $B_1 B_2 \ldots B_n$. However, if $t \vDash d_{l+1}(s,t)$ then there is a path that can be partitioned into blocks $B'_1 B'_2 \ldots B'_n$ such that all of the blocks are finite except B'_n and $\forall i\,[v \in B'_i \Rightarrow v \vDash \mathfrak{C}_l(M,\text{first}(B_i))]$. Since every state in B_i satisfies $\mathfrak{C}_l(M,\text{first}(B_i))$, the inductive hypothesis and the definition of E_l gives $B_i E_l B'_i$. We can also divide the infinite blocks B_n and B'_n into an infinite set of blocks containing one state each. Therefore, this path from t corresponds to the path from s, so we have a contradiction. We conclude that $t \nvDash d_{l+1}(s,t)$.

Now, these $d_{l+1}(s,t)$ describe the existence or nonexistence of a single path along which some $\mathfrak{C}_l$ formulas hold. By the definition of $s E_{l+1} v$, every path from s has a corresponding path from v along which the same $\mathfrak{C}_l$ formulas hold and vice versa. Therefore, $s E_{l+1} v \Rightarrow v \vDash d_{l+1}(s,t)$.

Therefore, the lemma is true. □

Theorem 15: Given a Kripke structure M with initial state s_0, there is a CTL formula $F(M,s_0)$ that characterizes that structure up to E-equivalence with respect to stuttering, i.e. $M', s'_0 \vDash F(M,s_0) \Leftrightarrow s_0 E s'_0$.

Proof: For any state s in M, let $s_1, \ldots, s_p$ be the *extended successors* of s, where an extended successor is a state that is not E-related to s and is reachable from s along a path consisting entirely of states that are E-equivalent to s. Next, we construct $G(M,s)$, which describes all of the transitions from s in M. In this construction, it is convenient to use the *weak until* operator, $A[f \mathrm{W} g] = \neg E[\neg g \mathrm{U} \neg f \wedge \neg g]$, which differs from the ordinary until in that it permits an infinite path along which every state satisfies the first argument. So now:

$$G(M,s) = \begin{cases} \bigwedge_i E[\mathcal{C}(M,s)\ \mathrm{U}\ \mathcal{C}(M,s_i)] \wedge A[\mathcal{C}(M,s)\ \mathrm{W} \bigvee_i \mathcal{C}(M,s_i)] \wedge \mathrm{EG}\ \mathcal{C}(M,s) & \text{if } s \models \mathrm{EG}\ \mathcal{C}(M,s) \\ \bigwedge_i E[\mathcal{C}(M,s)\ \mathrm{U}\ \mathcal{C}(M,s_i)] \wedge A[\mathcal{C}(M,s)\ \mathrm{W} \bigvee_i \mathcal{C}(M,s_i)] \wedge \neg\mathrm{EG}\ \mathcal{C}(M,s) & \text{otherwise} \end{cases}$$

Let $F(M,s_0)$ be the formula $\mathcal{C}(M,s_0) \wedge \bigwedge_s \mathrm{AG}\ (\mathcal{C}(M,s) \Rightarrow G(M,s))$.

The correctness of $F(M,s_0)$ is an easy consequence of the next two lemmas and theorem 13. □

Lemma 16: $s \models F(M,s)$.

Lemma 17: If $s \models F(M,t)$ and $s' \models F(M,t)$, then $s E s'$.

Proof of Lemma 16: Since every state is trivially equivalent to itself, $s \models \mathcal{C}(M,s)$ is true by lemma 14. Therefore, if $s \not\models F(M,s)$ then there is a $t \in M$ such that $s \models \mathrm{EF}(\mathcal{C}(M,t) \wedge \neg G(M,t))$. Let v be the state reachable from s that satisfies $\mathcal{C}(M,t) \wedge \neg G(M,t)$. By lemma 14, this condition implies $t E v$, so t and v must satisfy the same CTL formulas (theorem 13). We will show that $t \not\models \neg G(M,t)$, giving a contradiction. There are four cases.

1. $t \not\models E[\mathcal{C}(M,t)\ \mathrm{U}\ \mathcal{C}(M,w)]$, for some extended successor of t, w. By the definition of extended successor, there is a path from t to w and the states on this path are E-related to t. By lemma 14, these states must satisfy $\mathcal{C}(M,t)$. Since $w \models \mathcal{C}(M,w)$ is trivial, this path satisfies $\mathcal{C}(M,t)\ \mathrm{U}\ \mathcal{C}(M,w)$, which is a contradiction.

2. $t \not\models \mathrm{EG}\ \mathcal{C}(M,t)$. Since $\mathrm{EG}\ \mathcal{C}(M,t)$ is a conjunct of $G(M,t)$ if and only if $t \models \mathrm{EG}\ \mathcal{C}(M,t)$, we have an immediate contradiction.

3. $t \not\models \neg\mathrm{EG}\ \mathcal{C}(M,t)$. Since $\mathrm{EG}\ \neg\mathcal{C}(M,t)$ is a conjunct of $G(M,t)$ if and only if $t \not\models \mathrm{EG}\ \mathcal{C}(M,t)$, we have an immediate contradiction.

4. $t \not\models A[\mathcal{C}(M,t)\ \mathrm{W} \bigvee_i \mathcal{C}(M,w_i)]$. In this case, $t \models E[\mathcal{C}(M,t)\ \mathrm{U}(\neg\mathcal{C}(M,t) \wedge \bigwedge_i \neg\mathcal{C}(M,w_i))]$. Let $tt_1 \ldots t_n$ be this path, where $t_n \models \neg\mathcal{C}(M,t) \wedge \bigwedge_i \neg\mathcal{C}(M,w_i)$ and $\forall i < n\, [t_i \models \mathcal{C}(M,t)]$. By lemma 14, $\neg(t_n E t)$ and $\forall i < n\, [t_i E t]$. Therefore, t_n is an extended successor of t. But since $t_n \models \mathcal{C}(M,t_n)$ is trivially true, $t_n \models \bigwedge_i \neg\mathcal{C}(M,w_i)$ cannot be true, so we have a contradiction.

Therefore, the lemma is true. □

Proof of Lemma 17: Since sEs' if and only if sE_ls' for all $l \geq 0$, we will prove $s \models F(M,t)$ and $s' \models F(M,t)$ implies sE_ls' by induction on l.

Basis ($l=0$): Since $s \models F(M,t)$, $s \models \mathcal{C}(M,t)$ and therefore $s \models \mathcal{C}_0(M,t)$. Similarly, $s' \models \mathcal{C}_0(M,t)$, so $\mathcal{L}(s) = \mathcal{L}(t) = \mathcal{L}(s')$. Therefore, sE_0s'.

Induction: Assume that the result is true for l. We will now show it for $l+1$.

We want to show that every path, π, from s has a corresponding path, π' from s'. (The proof of the dual is identical.) We will use induction on the length of π to prove the slightly stronger result:

If $|\pi| \leq n$, then there is a corresponding path π' such that for some $v \in M$, last(π) $\models F(M,v)$ and last(π') $\models F(M,v)$.

Basis ($|\pi|=1$): In this case, $\pi = \langle s \rangle$. Let $B_1 = \langle s \rangle$ and $\pi' = B_1' = \langle s' \rangle$. By the outer inductive hypothesis, $s \models F(M,t)$ and $s' \models F(M,t)$ imply sE_ls', so $B_1 \, E_l \, B_1'$. Therefore, the paths correspond. Since the last states of each path satisfy $F(M,t)$, the base case is true.

Induction: Assume the result for $|\pi| \leq n$. Suppose that $\pi = ss_1s_2 \ldots s_n$, a path of length $n+1$. Now, $ss_1s_2 \ldots s_{n-1}$ is a path of length n, so by the inner inductive hypothesis, there is a corresponding path π' such that last(π') $\models F(M,v)$ and $s_{n-1} \models F(M,v)$ for some $v \in M$. Let $B_1B_2 \ldots B_m$ and $B_1'B_2' \ldots B_m'$ be the partitions that show that these paths correspond. There are three cases.

1. $s_n \not\models \mathcal{C}(M,v)$. Since $s_{n-1} \models F(M,v)$, we can infer that $s_{n-1} \models \mathrm{A}[\mathcal{C}(M,v) \ \mathrm{W} \bigvee_i \mathcal{C}(M,w_i)]$, where w_i are the extended successors of v. Since $s_{n-1}s_n$ is a path and s_n that doesn't satisfy $\mathcal{C}(M,v)$, we conclude that there must be an extended successor of v, x, such that $s_n \models \mathcal{C}(M,x)$. Since s_n is a successor of s_{n-1}, it must satisfy all of the AG formulas that s_{n-1} satisfies, so $s_n \models F(M,x)$.

From last(π') $\models F(M,v)$ we can infer that last(π') $\models \mathcal{C}(M,v) \wedge \mathrm{E}[\mathcal{C}(M,v) \ \mathrm{U} \ \mathcal{C}(M,x)]$. Therefore, there is a path $s_1's_2' \ldots s_k'$ where $s_1' =$ last(π'), $\forall i<k \, [s_i' \models \mathcal{C}(M,v)]$, and $s_k' \models \mathcal{C}(M,x)$. Now let $\pi = B_1 \ldots B_m\langle s_n \rangle$ and $\pi' = B_1' \ldots B_{m-1}'\langle B_m', s_2' \ldots s_{k-1}' \rangle\langle s'_k \rangle$. Since s_n and s_k' both satisfy $F(M,x)$, the outer induction hypothesis gives $\langle s_n \rangle E_l \langle s_k' \rangle$. Similarly, since the all the states in B_m, B_m', and $\langle s_2' \ldots s_{k-1}' \rangle$ satisfy $F(M,v)$, they are all E_l related to each other. Therefore, π and π' correspond with last(π) $\models F(M,x)$ and last(π') $\models F(M,x)$.

2. $s_n \models \mathcal{C}(M,v)$ and $v \models \mathrm{EG} \ \mathcal{C}(M,v)$. Since s_n must satisfy the same AG formulas as s_{n-1}, $s_n \models F(M,v)$. Now, last(π') $\models F(M,v)$, so last(π') $\models \mathrm{EG}\mathcal{C}(M,v)$. Therefore, last($\pi'$) must have a successor, s_1', which also satisfies $\mathcal{C}(M,v)$. Since this state must also satisfy all of the AG formulas, $s_1' \models F(M,v)$. Therefore, by the outer induction hypothesis, $s_nE_ls_1'$. So if we let $B_{m+1} = \langle s_n \rangle$ and $B_{m+1}' = \langle s_1' \rangle$, the paths correspond.

3. $s_n \models \mathcal{C}(M,v)$ and $v \not\models \mathrm{EG}\ \mathcal{C}(M,v)$. By the reasoning above, $s_n \models F(M,v)$, so $s_n E_l \mathrm{last}(B'_m)$. Therefore, π corresponds to π' with the same partition except that s_n is added to $B_{m'}$

We must also show that the blocks of the partitions are finite. The only problem is case 3, in which we might add an infinite number of states to a block of π. In this case, each of the states added to B_m satisfy $F(M,v)$, so if we add an infinite number of states to this block $\mathrm{first}(B_m) \models \mathrm{EG}\mathcal{C}(M,v)$ must be true. But since $\mathrm{first}(B_m) \models F(M,v)$, $\mathrm{first}(B_m) \models \neg\mathrm{EG}\mathcal{C}(M,v)$, so we have a contradiction. Therefore, all of the blocks of the partition must be finite.

Therefore, the lemma is true. □

Corollary 18: Given two structures M and M' with initial states s_0 and s'_0 respectively, $s_0 E s'_0$ if and only if for all CTL* formulas f without the nexttime operator, $M,s_0 \models f \Leftrightarrow M',s'_0 \models f$.

Corollary 19: Given two structures M and M' with initial states s_0 and s'_0 respectively, if there is a formula of CTL* without the nexttime operator that is true in one and false in the other, then there is also a formula of CTL without the nexttime operator that is true in the one and false in the other.

5. Algorithm For Stuttering Equivalence

In this Section we show how to compute the relation for equivalence with respect to stuttering for states within a single Kripke Structure M. The method that we suggest is polynomial in the number of states of M. To determine equivalence between states in two different Kripke structures M_1 and M_2, we form a Kripke structure M_{12} that is the disjoint union of these structures and check equivalence between the corresponding states in the combined structure.

We construct a relation C on $S \times S$ that is identical to the relation E defined in Section 4. $C = \bigcap_n C_n$ where C_n is defined as follows:

- $C_0 = \{(s,s') \mid \mathcal{L}(s) = \mathcal{L}(s')\}$

- In order to define C_{n+1} we must first define the set $\mathrm{NEXT}_{n+1}(s)$ of extended successors of s. We define this set in terms of the set $\mathrm{ST}_{n+1}(s)$ of *stuttering states* of s. $\mathrm{ST}_{n+1}(s) = \bigcup_k \mathrm{ST}^k_{n+1}(s)$ where,

    ○ $\mathrm{ST}^0_{n+1}(s) = \{s\}$

    ○ $\mathrm{ST}^{k+1}_{n+1}(s) = \mathrm{ST}^k_{n+1}(s) \cup \{s' \mid s' \notin \mathrm{ST}^k_{n+1}(s) \wedge \exists s'' \in \mathrm{ST}^k_{n+1}(s)[s'' \rightarrow s'] \wedge s' C_n s\}$

  $\mathrm{NEXT}_{n+1}(s) = \{s' \mid s' \notin \mathrm{ST}_{n+1}(s) \wedge \exists s'' \in \mathrm{ST}_{n+1}(s)[s'' \rightarrow s']\}$.

  We will also use a predicate $\mathrm{LOOP}_n(s)$ that is true iff there is a cycle containing only states in $\mathrm{ST}_n(s)$.

Now we can define C_{n+1} as follows:

$$\begin{aligned} C_{n+1} = \{(s,s') \mid & \mathrm{LOOP}_{n+1}(s) = \mathrm{LOOP}_{n+1}(s') \wedge s C_n s' \wedge \\ & \forall s_1 \in \mathrm{NEXT}_{n+1}(s) \exists s_1' \in \mathrm{NEXT}_{n+1}(s')[s_1 C_n s_1'] \wedge \\ & \forall s_1' \in \mathrm{NEXT}_{n+1}(s') \exists s_1 \in \mathrm{NEXT}_{n+1}(s)[s_1 C_n s_1'] \end{aligned}$$

Proof that the relation C constructed above is actually equal to the relation E defined in Section 4 will be given in the journal version of this paper. Since the inductive structures of the definitions of the two relations are different, it is necessary to split the proof into two parts: the first part shows that $E \subseteq C_i$ for every i; the second part shows that $C \subseteq E_i$ for every i.

Computing ST_n requires time $O(|S|^2)$. Computing C_{n+1} given C_n requires time $O(|S|^4)$, since at most $|S|^2$ pairs of states must be checked and each pair requires $O(|S|^2)$ time to check. The algorithm terminates as soon as $C_n = C_{n+1}$. Since at any previous step k, $|C_{k+1}| < |C_k|$ and since C_0 has at most $|S|^2$ pairs of states, there are at most $|S|^2$ steps in the construction of C. It follows that the complexity of the entire algorithm is $O(|S|^6)$.

If we replace each equivalence class of C by a single state, this algorithm can also be used to minimize the number of states in the structure.

6. Conclusion

The results of our paper have a number of surprising implications. For example, if a specification of a finite state concurrent program in CTL* is sufficiently detailed so that there is only one program (modulo one of our notions of equivalence) that meets the specification, then an equivalent specification could have been written in CTL instead. Another surprising consequence is that if a CTL* formula is not equivalent to any CTL formula, then it must have an infinite number of mutually inequivalent finite models. To see that this result is true, we first observe that since CTL* has the finite model property, it must be the case that if two CTL* formulas have the same finite models, they must have the same infinite models as well. Otherwise, if f_1 had an infinite model M that was not a model of f_2, $f_1 \wedge \neg f_2$ would have an infinite model, but no finite models, contradicting the finite model property of CTL* [5]. Therefore, we can characterize a CTL* formula by the set of *finite* models in which it is satisfied. If a CTL* formula is satisfied by only a finite number of equivalence classes of finite models, then the formula is equivalent to the disjuction of the CTL formulas that characterize the individual equivalence classes.

There are a number of directions for further research. First, from our construction, it appears that the characteristic formula of a Kripke structure might be quite large. It would be nice to have a lower bound on the size of this formula in terms of the size of the Kripke structure. Also, we conjecture that the $O(|S|^6)$ algorithm in Section 5 can be improved significantly. Finally, it would be interesting to see which of our results carry over to Kripke structures with fairness constraints, i.e. Büchi automata.

References

1. M. Ben-Ari, A. Pnueli, Z. Manna. "The Temporal Logic of Branching Time". *Acta Informatica 20* (1983), 207-226.

2. E.M. Clarke, E.A. Emerson. Design and Synthesis of Synchronization Skeletons using Branching Time Temporal Logic. Proceedings of the Workshop on Logic of Programs, Yorktown-Heights, NY, Lecture Notes in Computer Science #131, 1981.

3. E.M. Clarke, E.A. Emerson, A.P. Sistla. Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications: A Practical Approach. Tenth ACM Symposium on Principles of Programming Languages, Austin,Texas, 1983, pp. 117-126.

4. E.A. Emerson, J.Y. Halpern. "Sometimes" and "Not Never" Revisited: On Branching versus Linear Time Temporal Logic. Proceedings of the ACM Symposium on Principles of Programming Languages, Association for Computing Machinery, Austin, Texas, January, 1982. to appear in JACM.

5. E. A. Emerson and P. Sistla. Deciding Full Branching-time Logic. The Sixteenth Annual ACM Symposium on Theory of Computing, Association for Computing Machinery, Washington, D.C., May, 1984.

6. S. Graf and J. Sifakis. From Synchronization Tree Logic to Acceptance Model Logic. LNCS Vol. 193, Logics of Programs, 1985.

7. M. Hennessy and R. Milner. On Observing Nondeterminism and Concurrency. LNCS Vol. 85, 7th ICALP, 1980.

8. G.E. Hughes and M.J. Creswell. *An Introduction to Modal Logic.* Methuen and Co., 1977.

9. L. Lamport. "Sometimes" is Sometimes "Not Never". Seventh Annual ACM Symposium on Principles of Programming Languages, Association for Computing Machinery, Las Vegas, January, 1980, pp. 174-185.

10. L. Lamport. What Good is Temporal Logic? Proceedings of the International Federation for Information Processing, 1983, pp. 657-668.

11. O. Lichtenstein and A. Pnueli. Checking that Finite State Concurrent Programs Satisfy Their Linear Specification. Conference Record of the Twelth Annual ACM Symposium on Principles of Programming Languages, New Orleans, La., January, 1985.

12. R. Milner. *Lecture Notes in Computer Science.* Volume 92: *A Calculus of Communicating Systems.* Springer-Verlag, 1979.

13. B. Mishra and E. Clarke. "Hierarchical Verification of Asynchronous Circuits using Temporal Logic". *Theoretical Computer Science 38* (1985), 269-291 .

14. Z. Manna, P. Wolper. "Synthesis of Communicating Processes from Temporal Logic Specifications". *ACM Transactions on Programming Languages and Systems 6* (1984), 68-93.

15. A. Pnueli. Linear and Branching Structures in the Semantics and Logics of Reactive Systems. Proceedings of the 12th ICALP, 1985. Lecture Notes in Computer Science #194, Springer-Verlag.

16. A.P. Sistla, E.M. Clarke. "Complexity of Propositional Linear Temporal Logics". *Journal of the Association for Computing Machinery 32*, 3 (July 1985), 733-749.

17. M.Y. Vardi, P. Wolper. An automata-theoretic approach to automatic program verification. Logic In Computer Science, Cambrideg, Massachusetts, June, 1986 .

18. P. Wolper. Specification and Synthesis of Communicating Processes Using an Extended Temporal Logic. Ninth Annual ACM Symposium on Principles of Programming Languages, Association for Computing Machinery, Albuquerque, New Mexico, January, 1982, pp. 20-33.

END

FILMED

MARCH, 1988

DTIC